Práctica 1. Mejoramiento de la Imágen

Paul Sebastian Aguilar Enriquez, Carlos Ignacio Padilla Herrera y Simón Eduardo Ramírez Ancona

Resumen—Este documento presenta la implementación de un clasificador bayesiano para imágenes médicas en Matlab. Sc muestran imágenes del proceso, así como del resultado final. Se da un breve repaso acerca del filtro de media y se muestra el código fuente del clasificador implementado.

Index Terms—Reconocimiento de patrones, filtros, restauración, imágen, MATLAB.

1. Objetivos

El alumno:

1. Aprenderá a mejorar la imagen de acuerdo a las modificaciones que se le realizan originalmente.
2. Entenderá cómo aumentar las características de contraste y perfilado que apoyen a una mejor medición.

2. Introducción

Para la clasificación de datos existen muchas aproximaciones posibles. B = medfilt2(A) realiza un filtrado medio de la imagen A en dos dimensiones. Cada píxel de salida contiene el valor medio en un vecindario de 3 por 3 alrededor del píxel correspondiente de la imagen de entrada. medfilt2 almohadillas de la imagen con 0s en los bordes, por lo que los valores de mediana para los puntos dentro de la mitad de la anchura del vecindario ([m n]/2) de los bordes pueden aparecer distorsionados.

3. Desarrollo

3.1. A

Se tiene en el archivo de Midbrain.mat 2 imágenes a cargar. La primera es midbrain que es la imagen original.

A esta imagen se le realizó normalización entre 0 y 1, y se le añadió ruido sal y pimienta y a una segunda imagen se le agrego gaussiano. Un ejemplo es como se muestra en el código siguiente:

```
midbrainNoise=imnoise(midbrainnor,'salt & pepper');
```

A continuación se modificó el contraste con la siguiente instrucción:

```
for ii=1:ren
  for jj=1:col
    %get pixel value
    oldpixel=midbrainNoise(ii,jj);
      %check pixel value and assign new value
      if oldpixel<0.4
        new_pixel=(3/8)*oldpixel + (1/4)oldpixel;
      elseif (oldpixel>0.4 && oldpixel<0.6)
        new_pixel=oldpixel;
      else
        new_pixel = (3/8)*oldpixel +(3/8);
      end
      %save new pixel value in thresholded image
      midbrainthesh(ii,jj)=new_pixel;
  end
end
```

Quedando la función de modificación del contraste como:

$$y = (3/8)\ x + (1/4)x \quad \textbf{if } x \text{ in } [0.0\ ;\ 0.4]\ ,\ y > 0.40$$

$$y = x \quad \textbf{if } x \text{ in } [0.40\ ;\ 0.60] \quad (2)$$

$$y = (3/8)\ x + (3/8) \quad \textbf{if } x \text{ in } [0.6\ ;\ 1.0]$$

En realidad no servia la imagen asi por loque hay que aplicar el paso contrario a la misma:

1. Regresar a los valores iniciales de contraste
2. Quitar el ruido que se observa en la imagen. Usando un filtro para cada tipo de ruido, jugar con el tamaño de los filtros. 5x5, 9x9 y 11x11

Realice las operaciones necesarias para revertir las funciones.

3.2. B

Realizamos un filtrado para mejorar el procesamiento de la imagen y comparamos contra nuestra imagen anterior para ver si mejoro. Checamos los valores máximos y mínimos de la imagen original midbrain. Normalizamos

3.3. C

Genere un clasificador que le de una imagen binario (por regresión, por ejemplo) que separe el mesencéfalo del resto de su imagen. Si su respuesta fue un método de clasificación, investigue y mencione como lo programaría con otro método.

3.4. D

Todas las segmentaciones realizadas en imágenes médicas se utilizan para medir volúmenes, áreas y obtener parámetros importantes para los doctores. A partir de la imagen binaria haga un programa para medir longitudinal, y transversalmente el mesencéfalo. Y diga cuál es la longitud o el área que ocupa.

- *Pie*
- *Pie*

Referencia de pie

4. Resultados

Los resultados deberán presentarse con los cálculos respectivos.

5. Código fuente

```
close all
clear all
clc
%Se carga la imagen
im = imread('imagen.png');
%Se convierte la imagen a escala de grises
im_g = rgb2gray(im);
%Se filtra la imagen con un filtro de media en 2d
im_f = medfilt2(im_g);
%Se ajusta el contraste de la imagen
im_a = imadjust(im_f);
%Se obtiene el tamanio de la matriz de la imagen
[ncols,nrows]=size(im_a);
%Se despliega la imagen para obtener la posicion de
%los pixeles semilla
imshow(im_a)
%Se guardan los valores de los pixeles
[x,y] = getpts;
%Se convierte a valores enteros para usar como parametros
xi=int32(x);
yi=int32(y);
%Se cierra la figura creada
close all
%Se binariza la imagen. Calculando el umbral de imagen adaptable
%localmente elegido utilizando estadisticas de imagen
%de primer orden locales alrededor de cada pixel.
BW = imbinarize(im_a, 'adaptive');
%Se convierten los valores obtenidos a enteros.
bin = im2uint8(BW);
%Se crea una mascara binaria para los pixeles
%con intensidad de gris similar con una tolerancia de 1
bw = grayconnected(bin,xi,yi,1);
%Se obtiene el area total
area = bwarea(bw);
%Se recorta la imagen
im_r = imcrop(bw,[285,300,95,70]);
%Se muestran los resultados
figure
subplot(3,2,1),imshow(im_g),title('Original')
subplot(3,2,2),imhist(im_g,128),title('Histograma de original')
subplot(3,2,3),imshow(im_a),title('Filtrada y ajustada')
subplot(3,2,4),imhist(im_a,128),title('Histograma filtrado y ajustado')
subplot(3,2,5),imshow(bw),title('Mesencefalo')
subplot(3,2,6),imshow(im_r),title('Imagen recortada')
```

6. Conclusiones

The conclusion goes here.

El clasificador que implementamos es bastante sencillo de aplicar y entender, ya que solo emplea conceptos esenciales que devuelve resultados bastante aceptables para la clasificación de imágenes. Los resultados obtenidos con la imágen de prueba fueron bastante aceptables, ya que la mayoría de los pixeles están bien clasificados. Con algunos pequeños retoques es probable que la implementación sea capaz de tener un mejor desempeño.

Referencias

[1] W. Pratt, *Digital Image Processing*, John Wiley & Sons Inc, 2001.
[2] Gonzales Woods, *Digital Image Processing*, 2004.